



Classification and Workflow of ML systems



Outline



1. ML vrs traditional programming



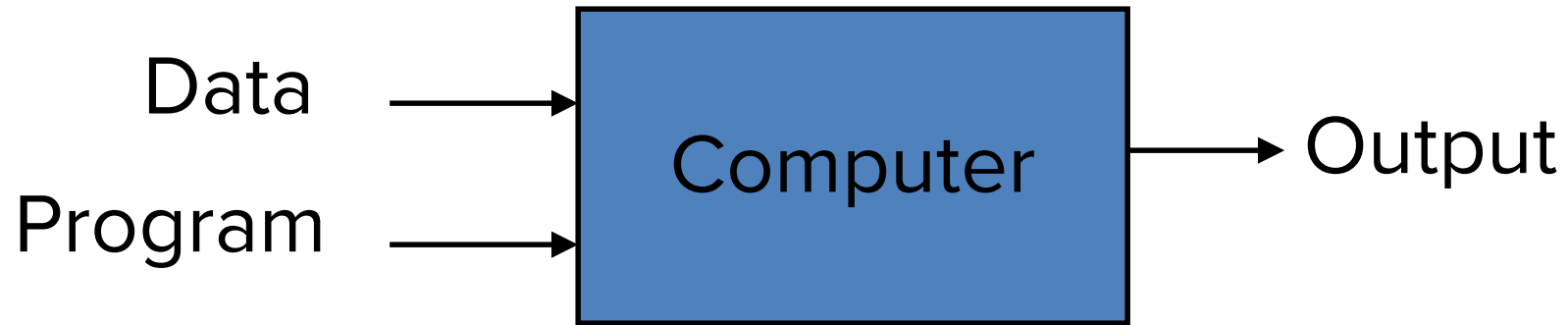
2. The design cycle of ML systems

Data selection, preparation and pre-processing

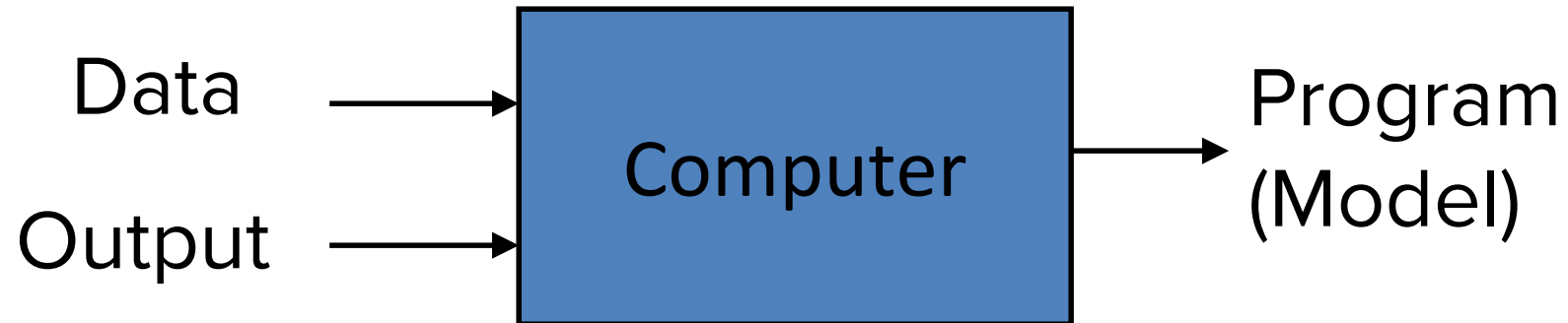
Model selection: types of ML algorithms

Model evaluation and fitting

Traditional Programming



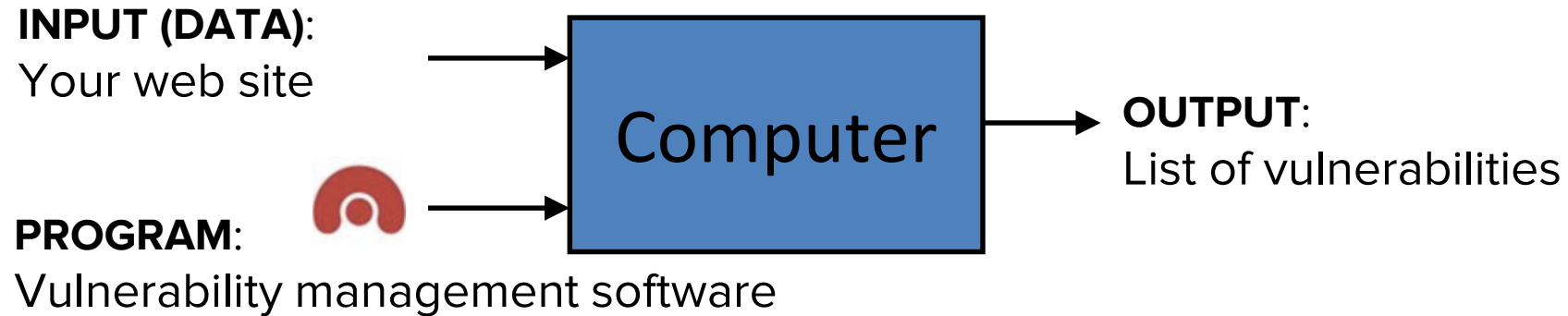
Machine Learning



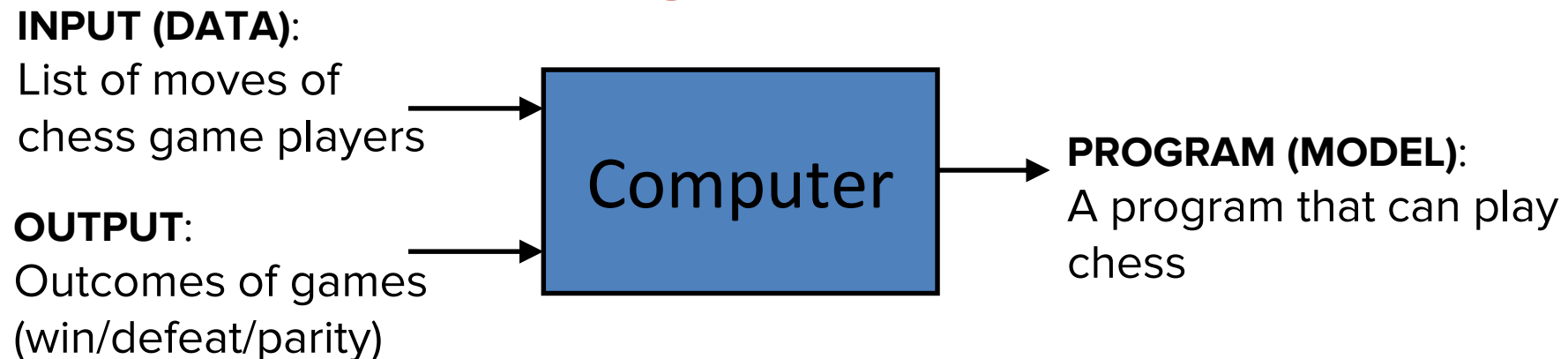
Def: «A computer model is **an abstract mathematic representations (some $f(x)$)** of a real-world event, system, behavior, or natural phenomenon (x). A computer model is designed to behave just like the real-life system. »

Example

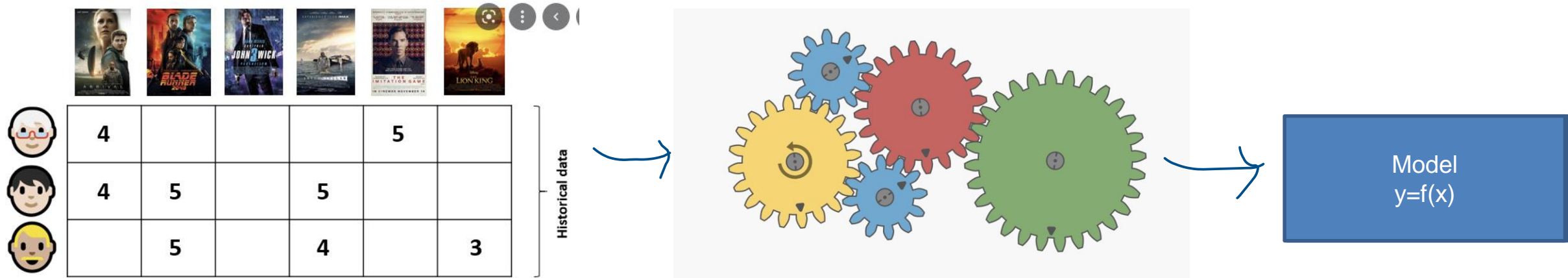
Traditional Programming



Machine Learning

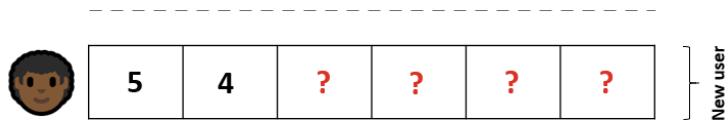


ML systems learn a (predictive, prescriptive, descriptive) **MODEL** either from examples (historical data) or from experience

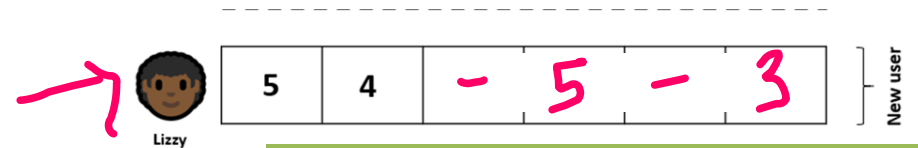


Historical data: tuple $\langle x_i, f(x_i) \rangle$

x_i : user, $y_i=f(x_i)$: his/her preference for a movie



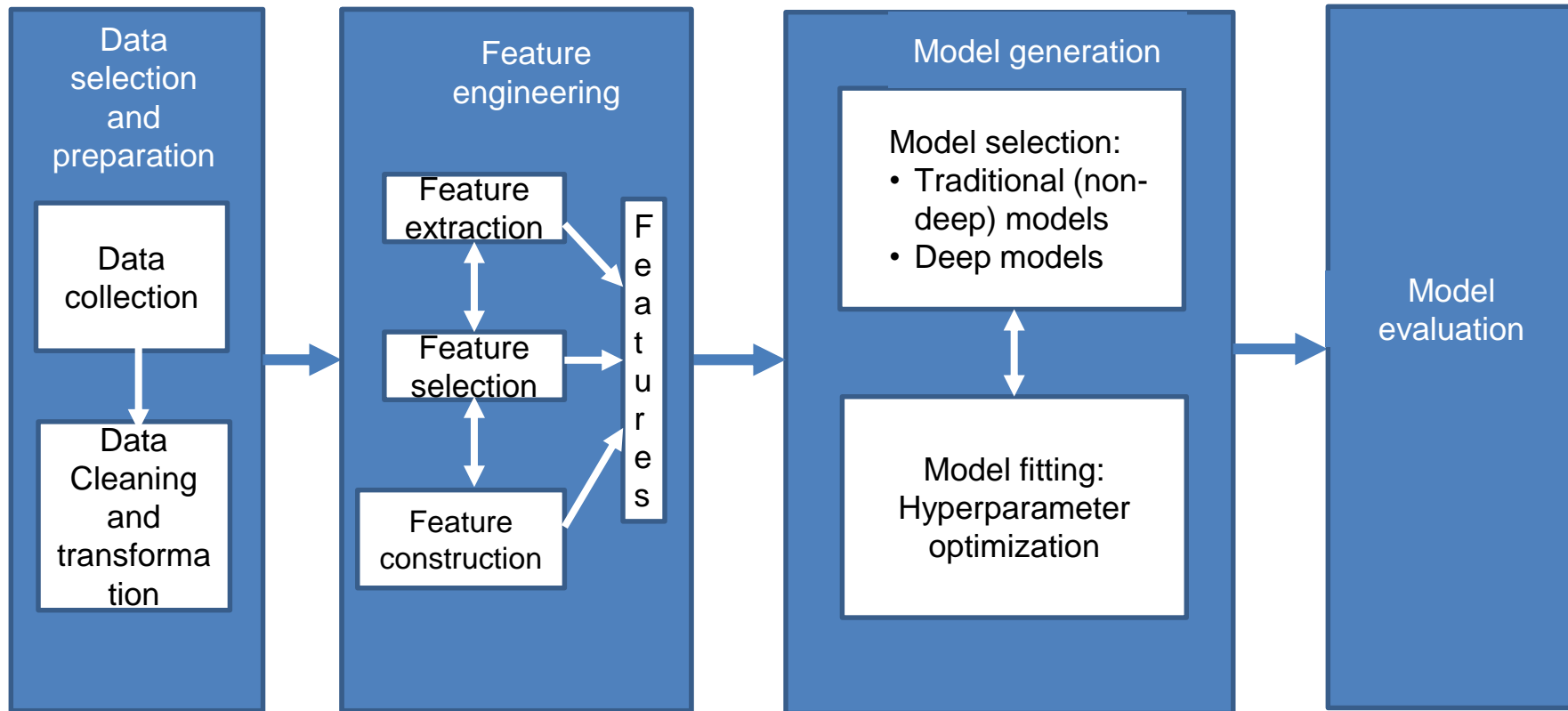
«x» here is a vector of integers with empty values



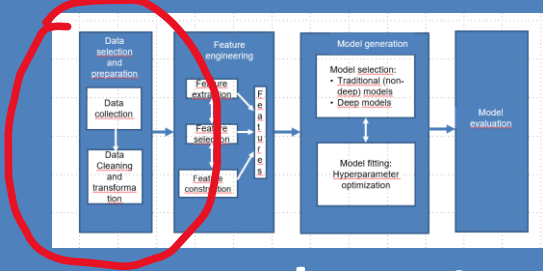
here is the input were empty values are filled

$f(x)$ is the function able to compute missing values

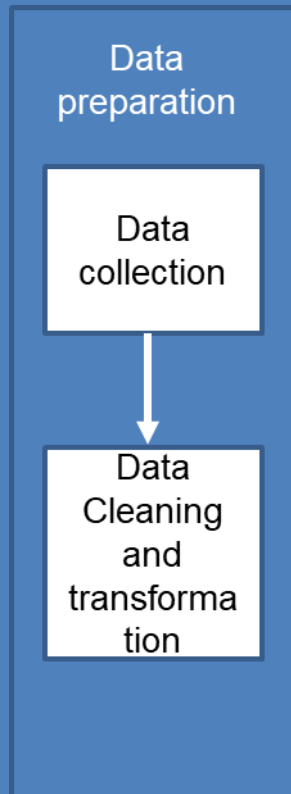
ML systems: Design cycle



Designing/selecting a ML algorithm (able to learn a «*model*») is just **one step** of a very complex pipeline



Data selection and preparation



Which data, what for?

Where do I find my data?

Are they ready for analysis? (usually NOT)

Which data, what for?

Data are the fuel of machine learning – the success of specific algorithms often depends on the data they use (e.g., convolutional neural networks are very effective on image data, long-short term memory LSTM are tuned for sequential data like sensor signals, sequences of events..)

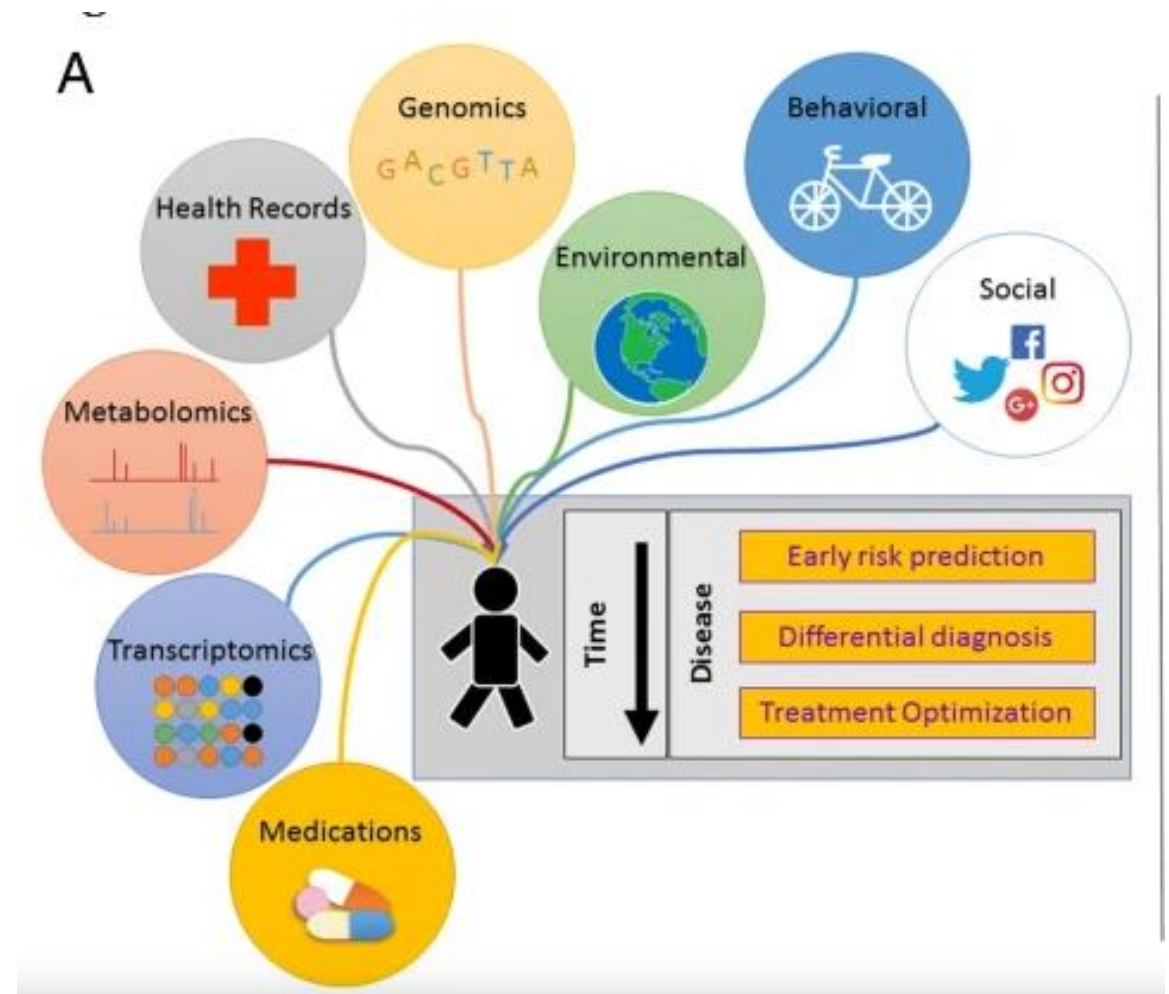
Not only the type of data matters, but also the quantity and quality, and **how we represent it (e.g., how do we represent a free text?)**

Often the same type of data can be represented in different ways (amenable for exploiting the features of different algorithms)

More often, heterogeneous types of data are needed (images, texts, tables..)

Example: medical machine learning

- Several types of data
 - Patients health records (sequences of diagnoses treatments clinical tests results)
 - Info on medications/prescriptions and their effect (tables or texts)
 - Behavioural data (signals from sensors)
 - Social data (texts, images)
 - Biological /genetic data (strings, networks)
 - Clinical images
- Several «formats»
 - Tables (csv files)
 - Graphs (matrixes)
 - Sequences (values with timestamps)
 - **Unstructured**, such as text, signals, videos, audio..
 - ...
- Other issues: privacy, representation standard, data quality, **ethics & bias**



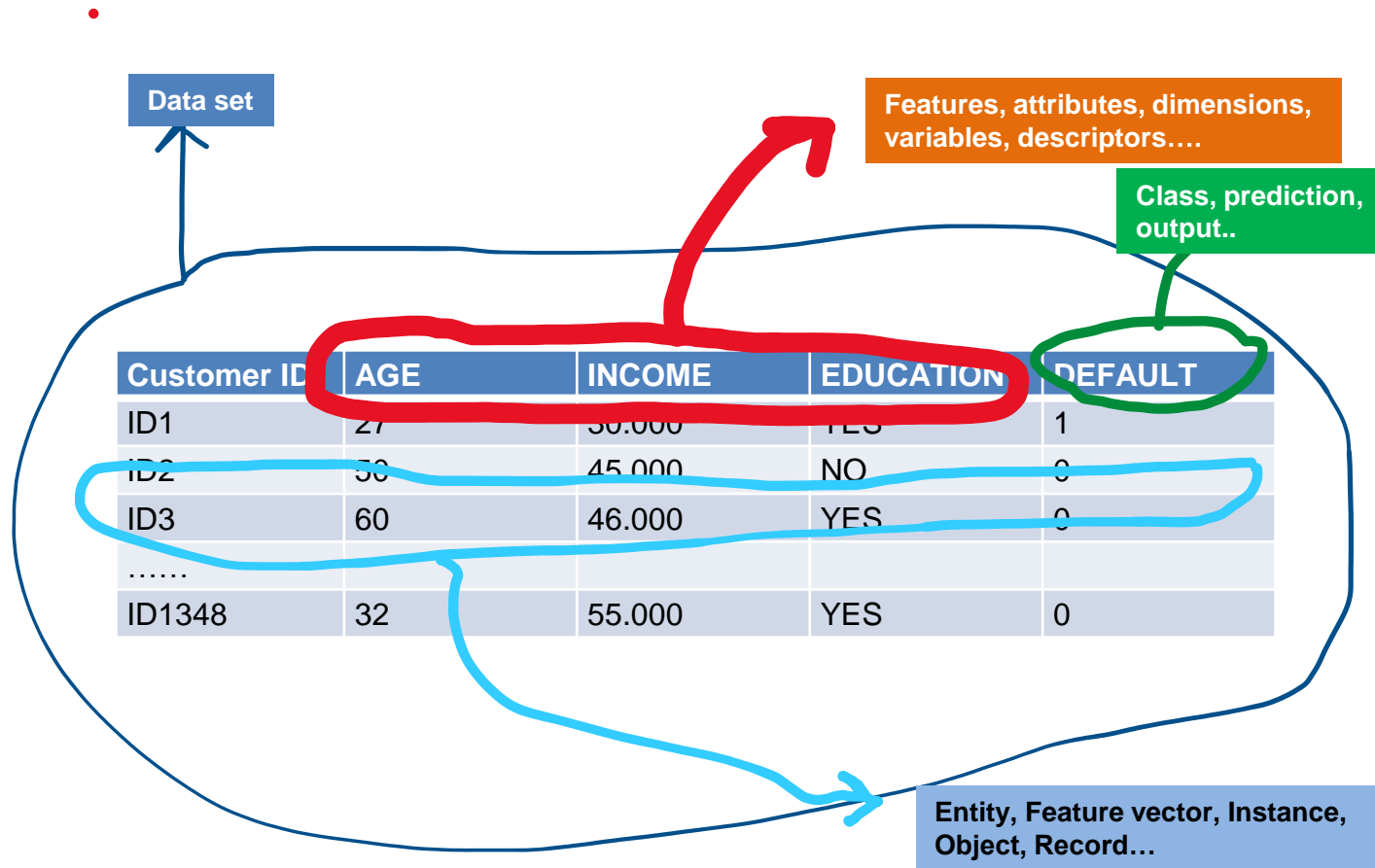
Where do I find my data?

- **Lots of publicly available and legacy data:**
 - **Social data** (social networks, blogs): to mine user opinions, trending topics, market forecasts
 - **Sensors data** (signals from devices e.g. vending machines, packages, wearable devices, sensor networks..): to detect anomalies (remember Magpie vaccines), learn trends..
 - **Clickstream data** (clicklogs of web sites): for traffic and e-commerce analysis
 - **Environmental data** (geolocations, meteorological data): to produce recommendations, supply chain, market forecasts..
 - **Images, videos, signals** (medical imaging, landscapes, portraits): to detect anomalies, security, fraud detection..
 - **Audio** (speech, sound): to mine opinions, fraud detection, environmental analysis

Are data ready for analysis? (formatting problems)

- Most of the time **they are NOT**:
 - not in a readily processable format, such as images, signals, text
 - Often heterogeneous data can be converted into a simplest form, such as tables
 - In some case, we can also process data «as they are» (e.g., pixels)
 - «[representation learning](#)» is about finding the best representation for our data (given the objectives of data analytics)
 - **Keep in mind**: images are one of the «best» types of data, and the one on which deep algorithms work best. More recently, NLP systems got on the podium. But the world is full of other types of data (speech, human-entered symbolic data, genetic data that have the form of complex graphs..), and they are dirty, incomplete, heterogeneous, inconsistent..

Tables (matrixes) are the «simplest» format – although very rarely data are available in this ready-to-use format



Example: (excerpt of) historical data on bank customers who applied for credit. The «DEFAULT» attribute tells us if they have been defaulters or non defaulters. ML task is learning to predict future defaulters to inform about the risk of granting a credit to a new customer.

Are data ready for analysis? (cleaning and transformation)

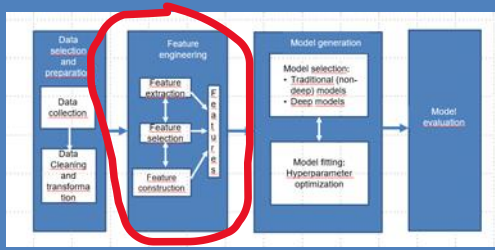
Data may need a lot of cleaning
(transformations):

- Get rid of errors, noise, missing values
- Removal of redundancies
- Check for bias and inconsistencies

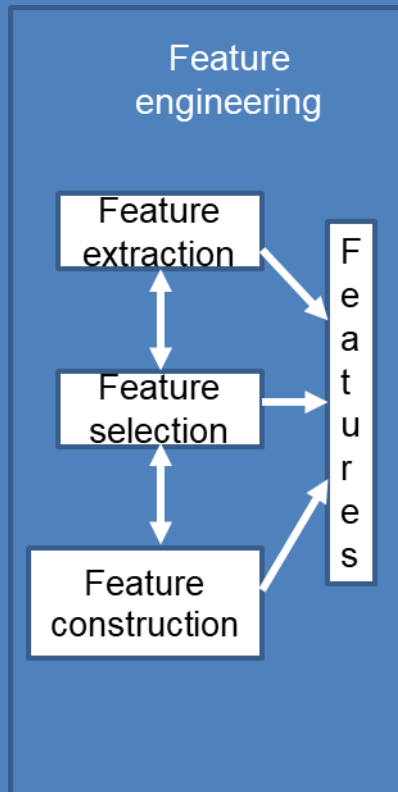
Data may need a lot of preprocessing:

- Renaming
- Rescaling
- Normalization
- Discretization
- Abstraction
- Aggregation
- New attributes (augmentation)

Will see later (data preparation pipeline)
examples of data preprocessing



Feature engineering



The descriptors (features) of our data can be too many (e.g. the words of a document archive) in general, or for the objectives of our analysis; but they can also be too few

Feature identification and engineering is about finding the best descriptors

To reduce the number of features, we can use **dimensionality reduction**: algorithms to **compress** low level features into higher level «semantic» descriptors (e.g., from words to sentiment, from words to «meanings»..)


But we may also want to **augment** our data: so-called generative methods can be used for this purpose. For example, we have too few patient records for, say, some rare disease and we want to generate «realistic» synthetic data


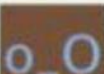

Will see all that later

Example (feature identification)

Task: determine the “sentiment” of Twitter messages – we don’t need to consider all words!

SEARCHED TERM	POSITIVE TWEETS	NEUTRAL TWEETS	NEGATIVE TWEETS	TOTAL TWEETS
taxes	154	1272	77	1500

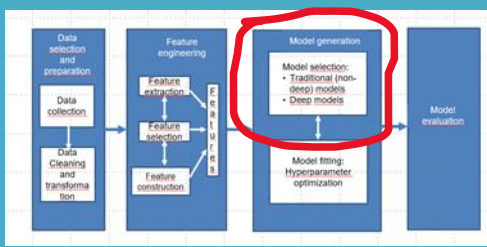
**Doug Ford** @fordnation · 7h
I'll be joining my friend @jkenney in Alberta next week. Jason has been working tirelessly to fight the federal carbon tax, and I am proud to say that Ontario will stand with Albertans who oppose this unfair and burdensome tax on families and businesses. #abpoli #onpoli

10.07  law and econ = great class. learned abt coase and torts and injury (view)	 happens if AIG repays the loans? Will the gov refund the taxes? #gawmal #tcot (view)	 those of you going maybe next year. feeling the need to paint something (view)
at least i am sitting in the sun preparing for taxes - that's the good news (view)	 South Fayette to hold the line on property taxes: South Fayette school administrators last night proposed a \$31... http://tinyurl.com/c6vnms (view)	my speck package came today.... i hate that i have to pay 30% taxes at the dutyoffice >:((view)
@johnchow if you have a friend into import/export.. you can :) you just need to pay for the import taxes and stuff.. :) but still cheap! :d (view)	 http://cliqz.com/us.headlines/c/9221.html : Congress Looking at New Taxes on AIG Bonuses (view)	yeah, i know aig execs are greedy, but there's something wrong when govts create new taxes just to hound a handful of people. (view)
 apparently, majority of those losing jobs are in the private	 South Fayette to hold the line on	@codeape_ did you just do your

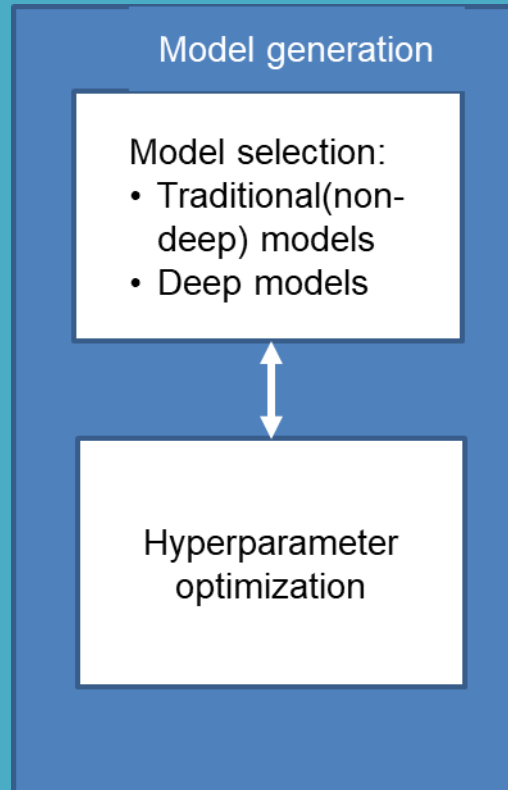
Example 2 (feature identification)



If the task is recognizing frogs vrs lions, we don't need representing data at the pixel level! A colour histogram would do..(**Occam razor**: *entia non sunt multiplicanda praeter necessitatem*)



Model selection



- Remember: model learning= devising an algorithm (implemented with a program) to learn a function (the model) $y=f(x)$ where x is an *instance* of input data
- What are the different types of ML systems?
- ML systems can be classified according to:
 - How they learn
 - What kind of function they learn

Classification of ML algorithms

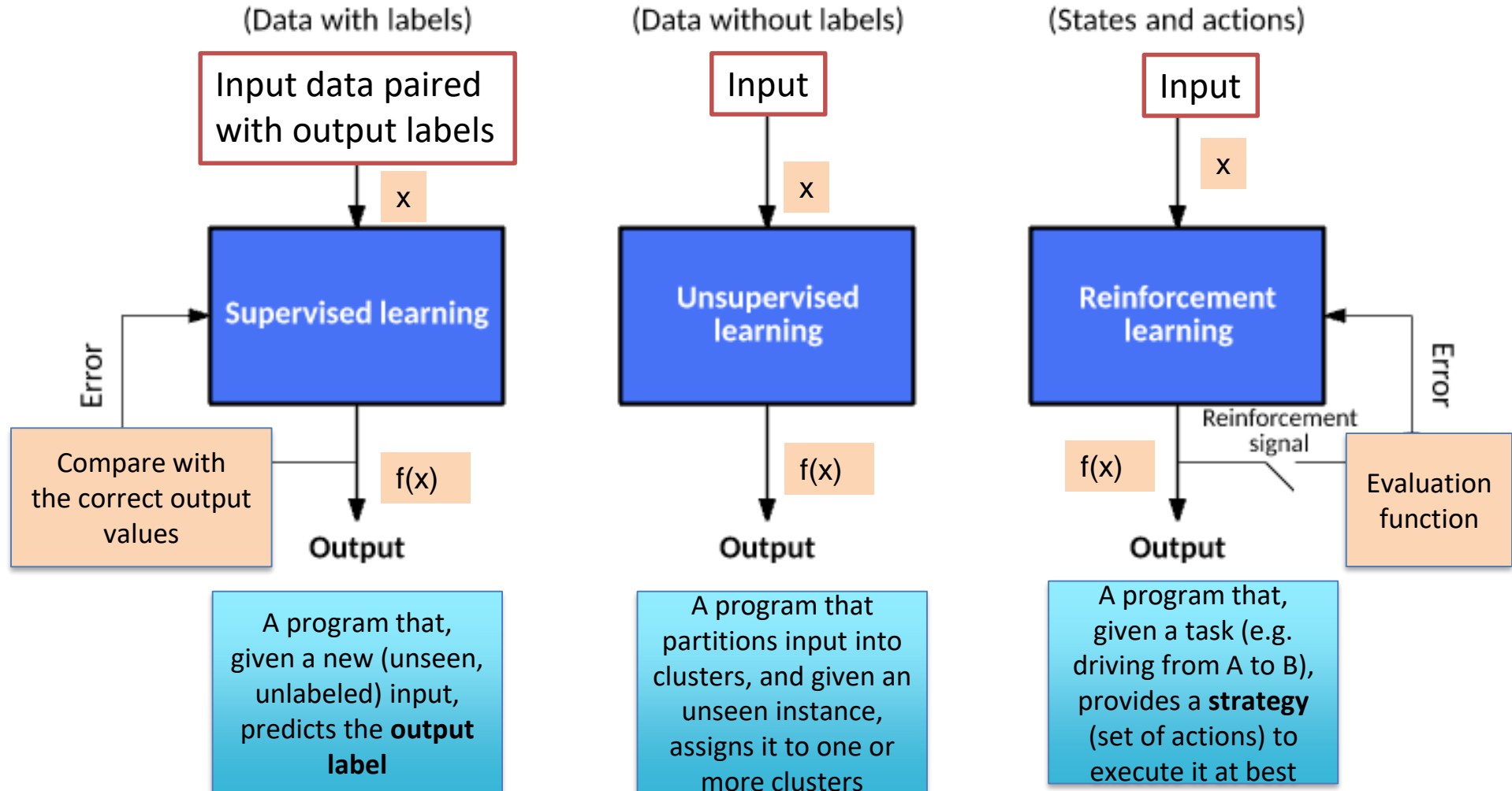
Types of learning (how
do we train?)

Types of
predictions **function** (the
model $f(x)$) we learn)

Classification of ML learning systems according to the training method

- Four categories of ML algorithms:
 - **Supervised (inductive) learning**
 - Learning from "teachers": Training data includes examples of (correct/desired) outputs
 - **Unsupervised learning**
 - Training data does not include output labels
 - **Semi-supervised learning**
 - Training data includes a few output labels
 - **Reinforcement learning**
 - Learning by doing: start with random strategies and adjust based on results wrt a stated objective

Supervised/Unsupervised/Reinforcement



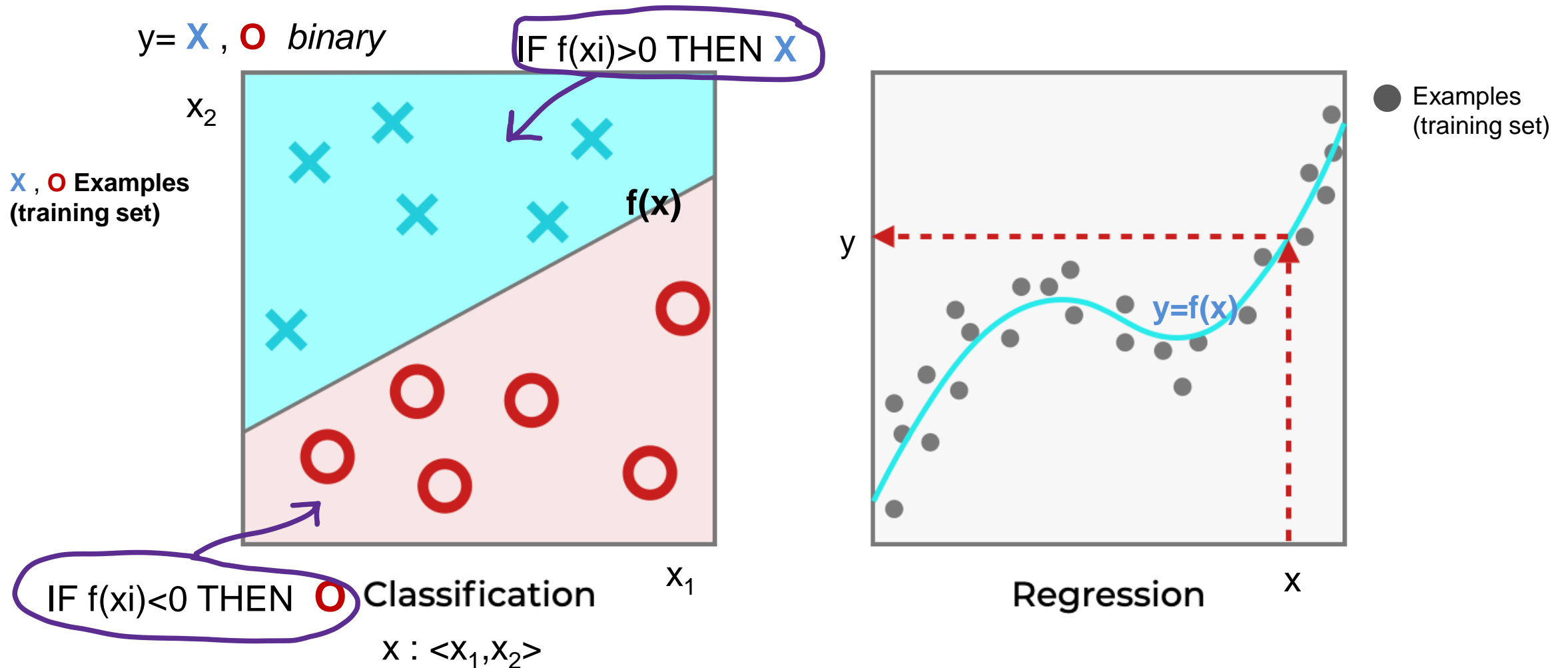
Classification
of ML systems
according to
the type of
learned
function
 $y=f(x)$

- Range of $f(x)$ is DISCRETE → Classifier
- Range of $f(x)$ is CONTINUOUS → Regressor
- Note: the **domain** of $f(x)$ can be discrete or continuous or both if x is multivariate ($\mathbf{x}: x_1, x_2 \dots x_n$)
- Furthermore:
 - $f(x)$ can be a logic, algebraic or probabilistic function

Classifiers/regressors

- **Classifier:** given input data, the system learns a categorical (e.g., boolean) function. For example, given a (vectorial) representation of a patient, returns 0 if the patient is predicted not at risk of cardiovascular event, 1 is at risk
- **Regressor:** given input data, the system returns real value(s). For example, given a sequence of stock market values at time t_1 t_2 .. t_n , it is able to predict the value at t_{n+1}

Regression vs Classification, Explained



(more on) Classifiers vrs Regressors

Output : DISCRETE

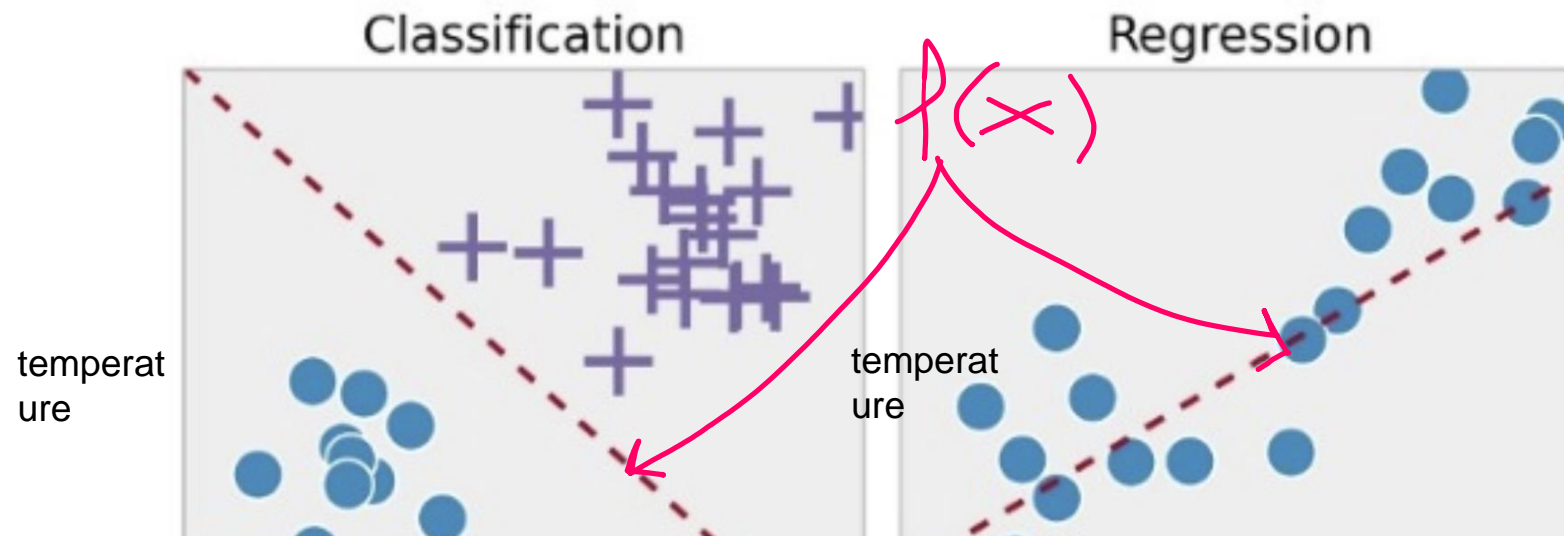
Objective: find **decision boundary** $f(x)$

Example: will the weather tomorrow be CLEAR or CLOUDY?

Output : CONTINUOUS

Objective: find **best-fit** curve $f(x)$

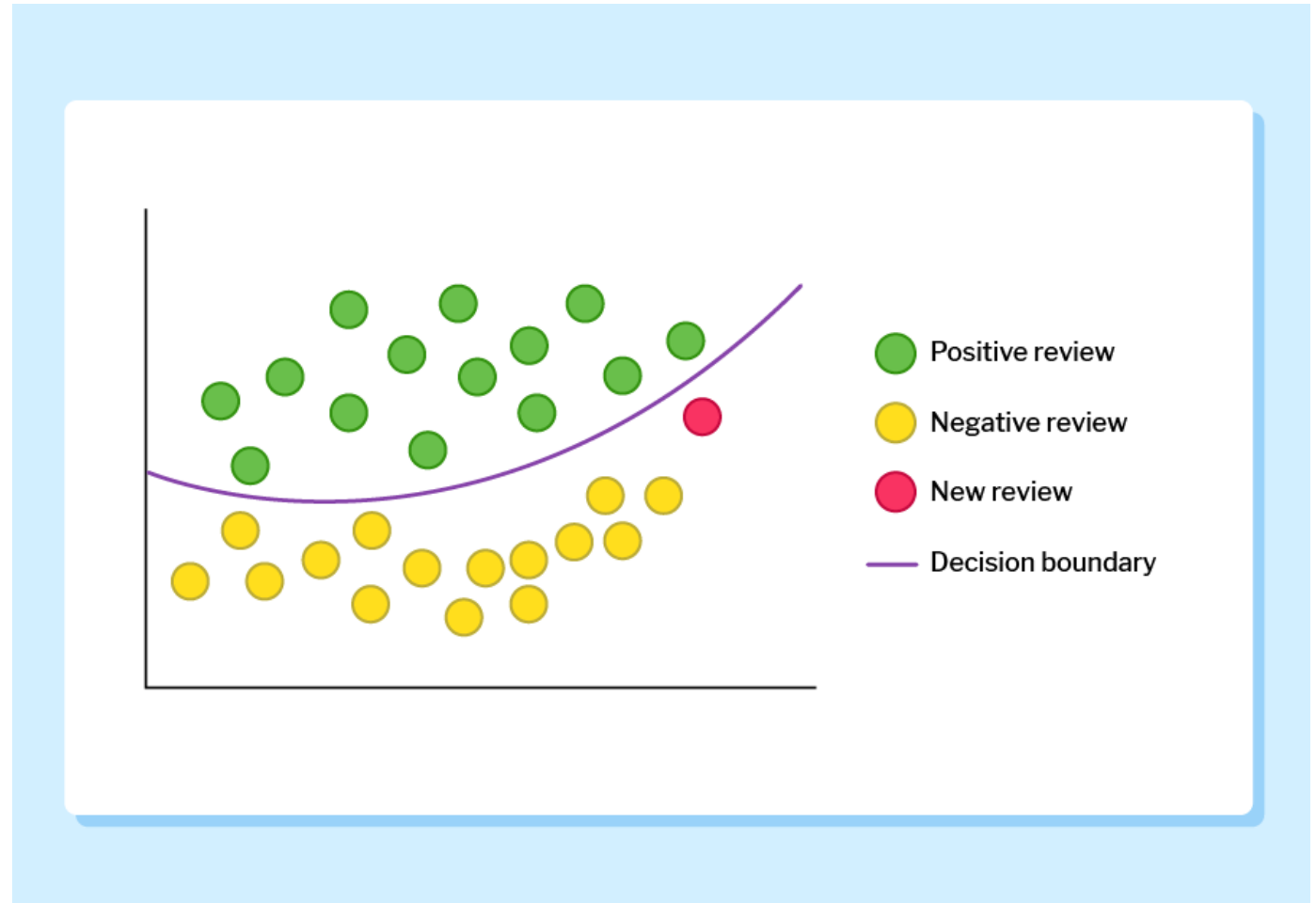
Example: what will be the temperature tomorrow?



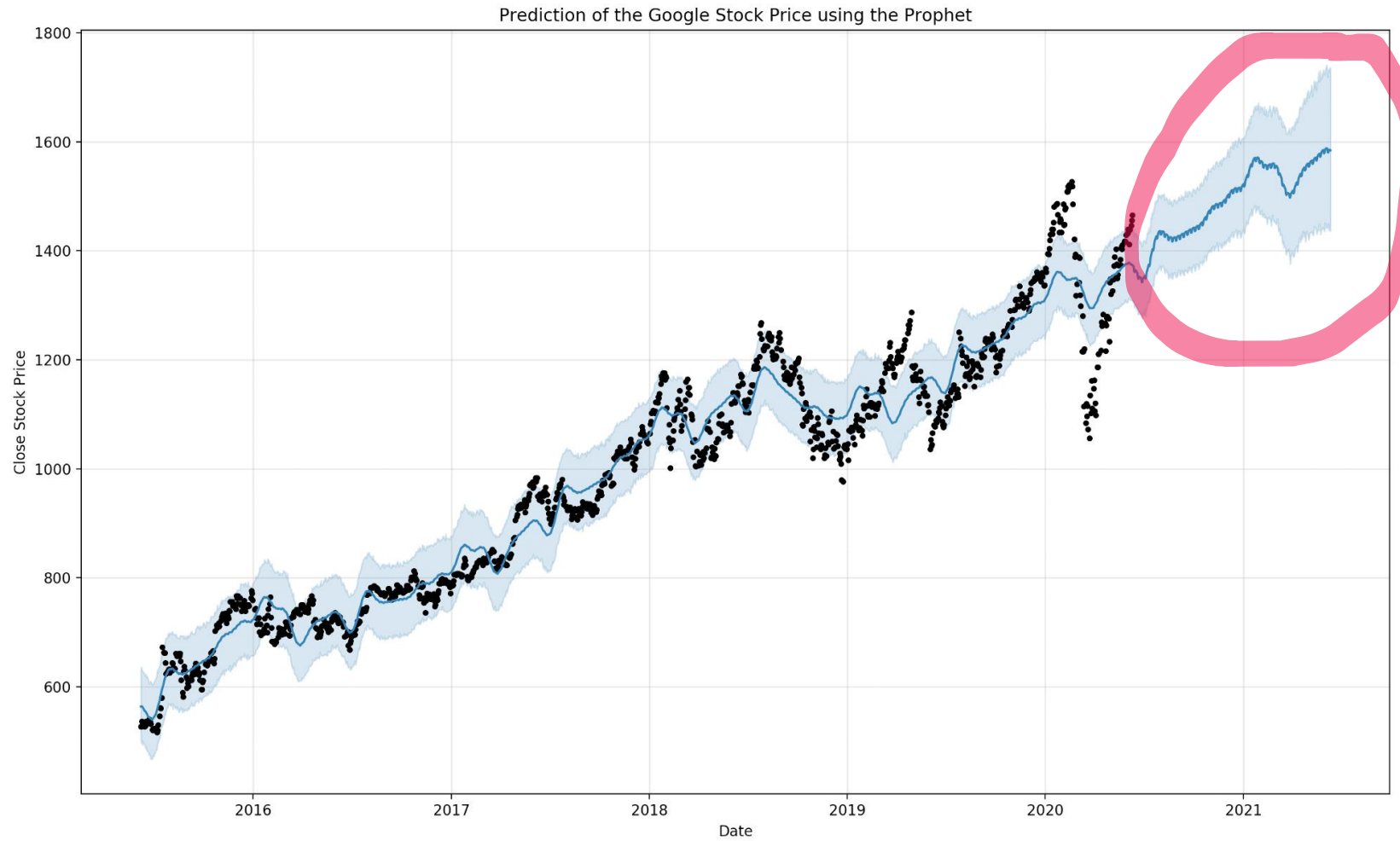
The term «predictor» indicates either a classifier or a regressor

Example of binary classifier: movie reviews classification

- Points (training examples) are some geometric representation of a movie review (e.g. x =number of positive adjectives in the review and y = number of emoticons in the review).
- Colors are the classification labels (positive or negative) in the training data
- Once a model is learned (the decision boundary $f(x)$, purple line), a new «point» (e.g., the red point in figure) is classified depending on its position wrt the boundary line (below \rightarrow negative, above \rightarrow positive)



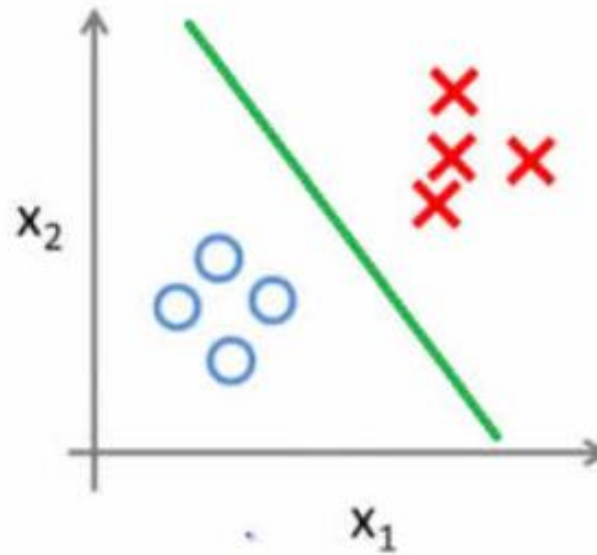
Financial forecasting: regressors



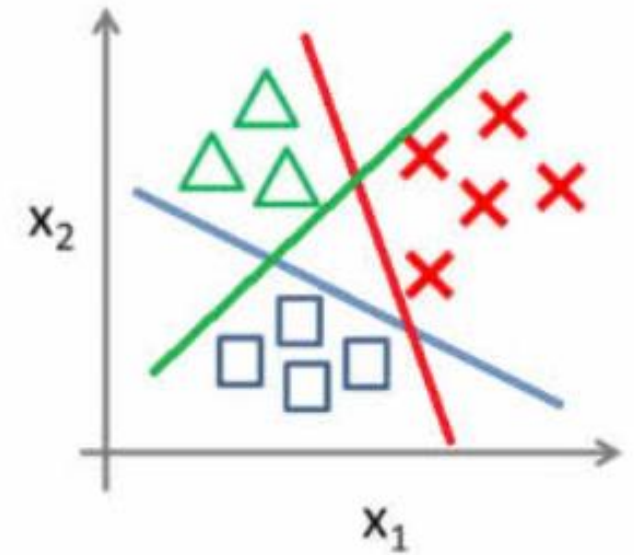
Input are the price values of a stock (black points), need to learn a function $y=f(x)$ that predicts *future values*. The **blue line is the learned $f(x)$** , trained over historical data (black dots); the light blue strip model the uncertainty. The blue line is «fitted» using past data, and then can be used to predict the future trend (circled area)

Binary vrs Multi-class classification

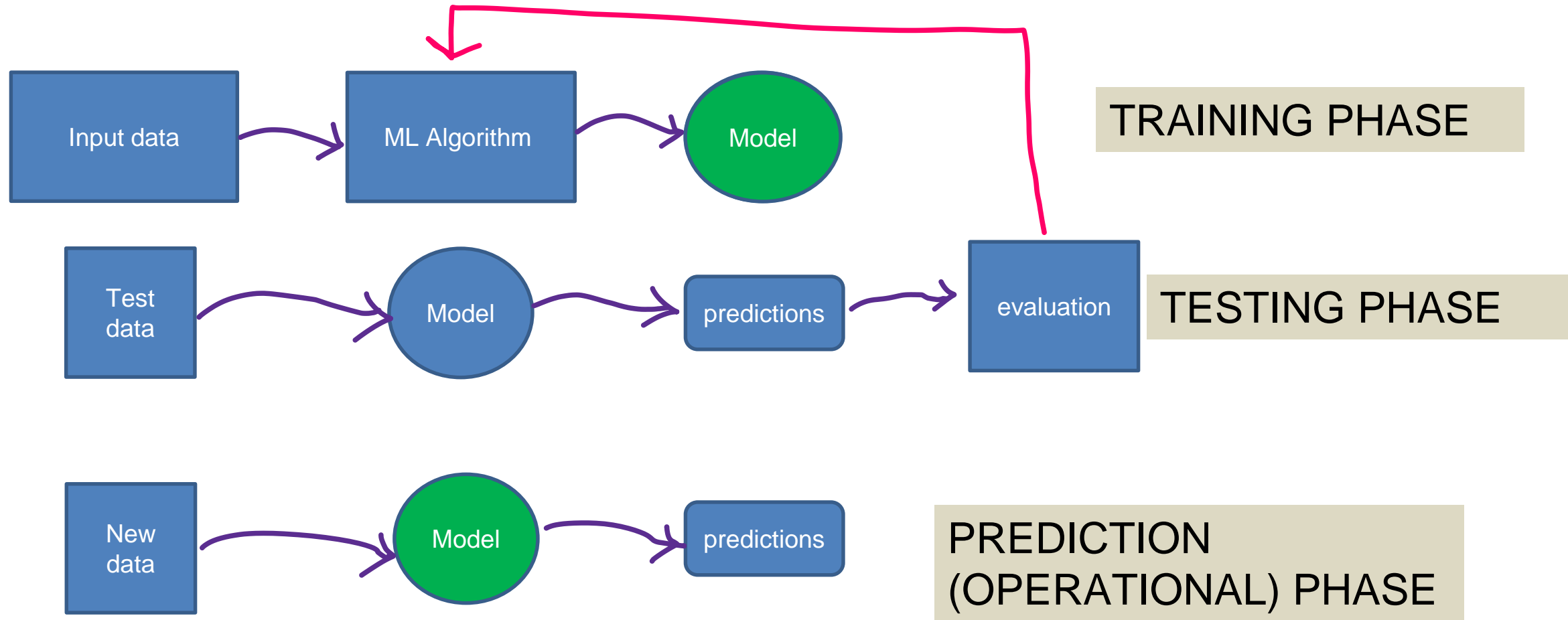
Binary classification:



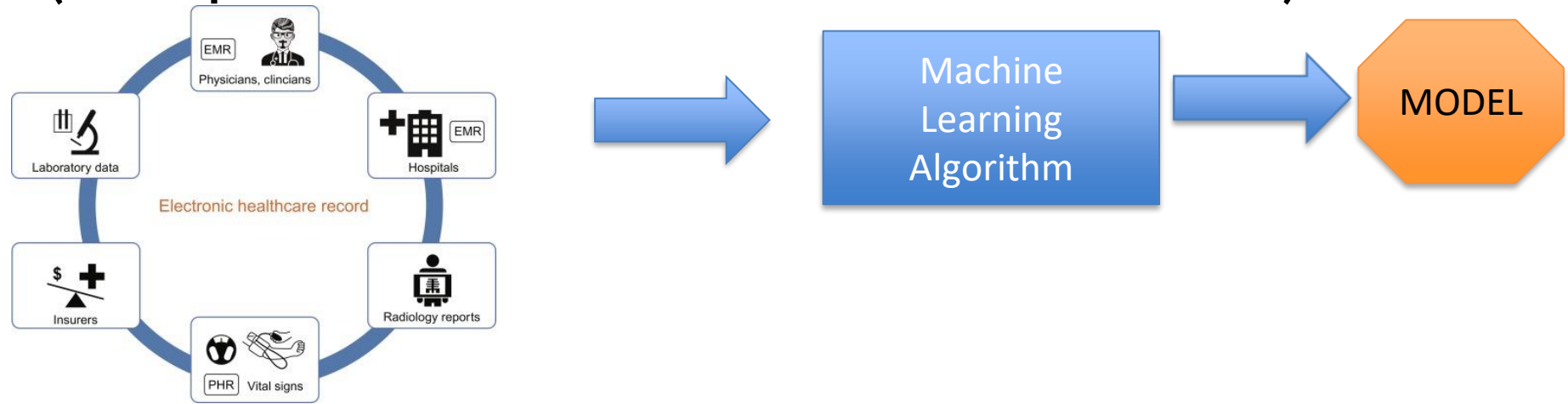
Multi-class classification:



The 3 phases of supervised ML



Example: cardiovascular risk prediction (is a patient at risk of a heart attack?)

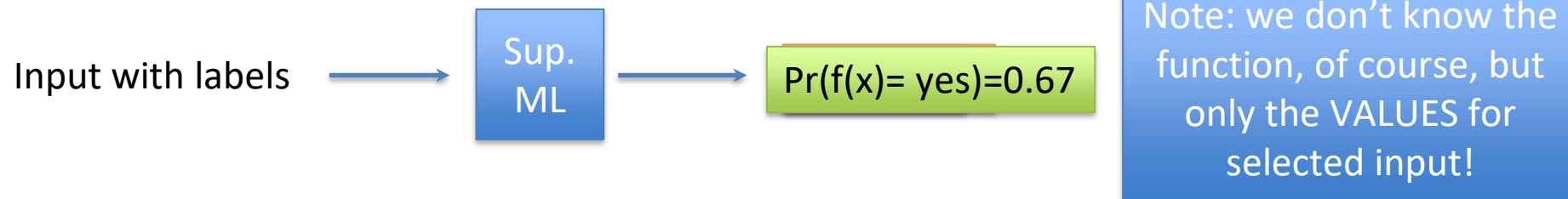


During the **training** phase, a machine learning algorithm use historical data on past patients' visits and diagnoses (the training set) to learn a classification **model**.

During the **prediction** phase, the model *computes* a **function** $\mathbf{f}(\mathbf{x})$ that, given an instance \mathbf{x} (in our case, a patient with its descriptors), generates a label $\mathbf{y} = \mathbf{f}(\mathbf{x})$ for example, a risk probability

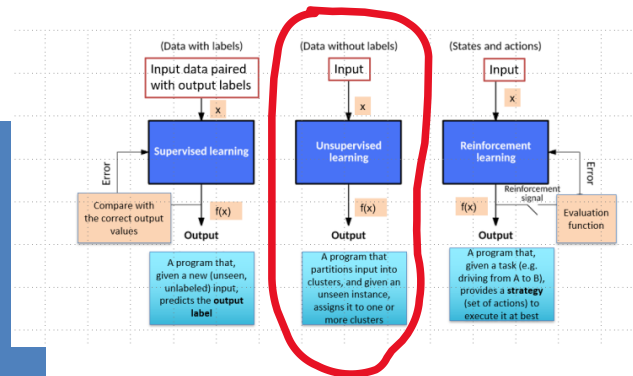
Supervised Learning: formal definition

- **Given** examples (x_i, y_i) of an **unknown** function $y = f(x)$
Learn $f(x)$ to predict its values for unseen examples x
 - If $f(x)$ is **discrete**: **Classification Problem**
 - If $f(x)$ is **continuous**: **Regression Problem**
 - If $f(x)$ represents a **Probability** (continuous and $0 \leq y \leq 1$):
Probability Estimation

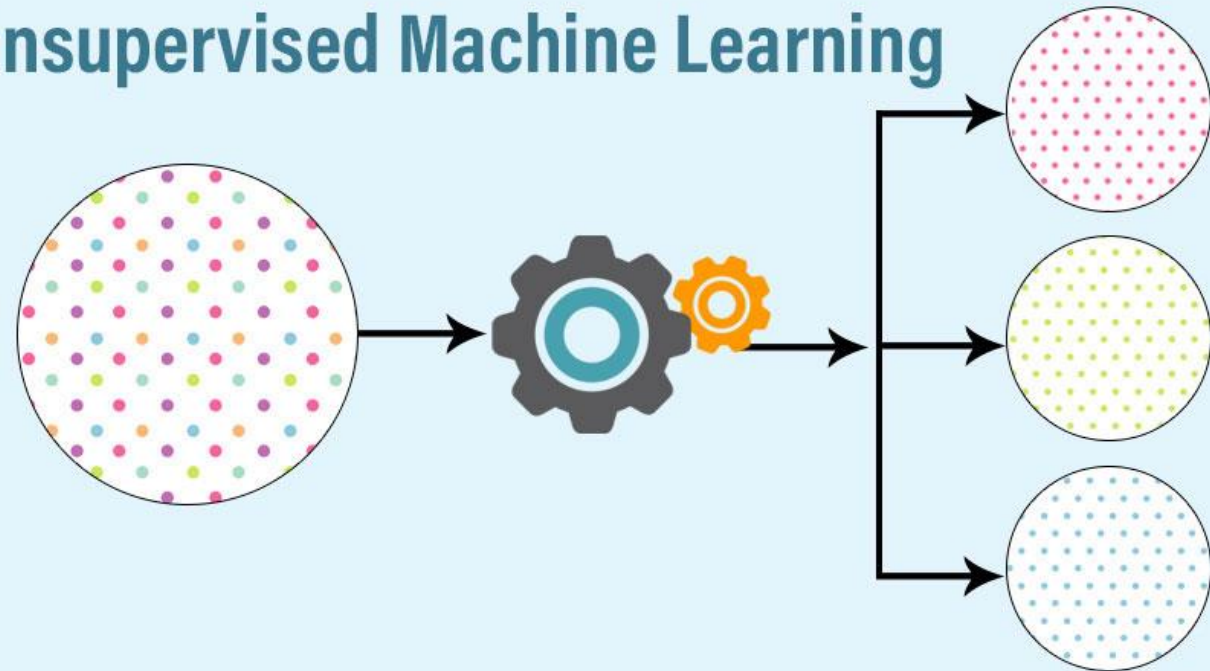


Unsupervised ML

- In unsupervised learning, **data are unlabeled**. No “ground truth” is provided for the output. One common type of unsupervised learning is **clustering**. The algorithm must learn a model to group instances according to their “similarity”. Groups are called “clusters”.
- During the training phase, the system learns the clusters (model)
- During the prediction phase, the system assigns any new, unseen instance to a cluster (output is a class label, or a probability)
- **Problem is testing:** no ground truth, so usually a number of cluster quality measures are used

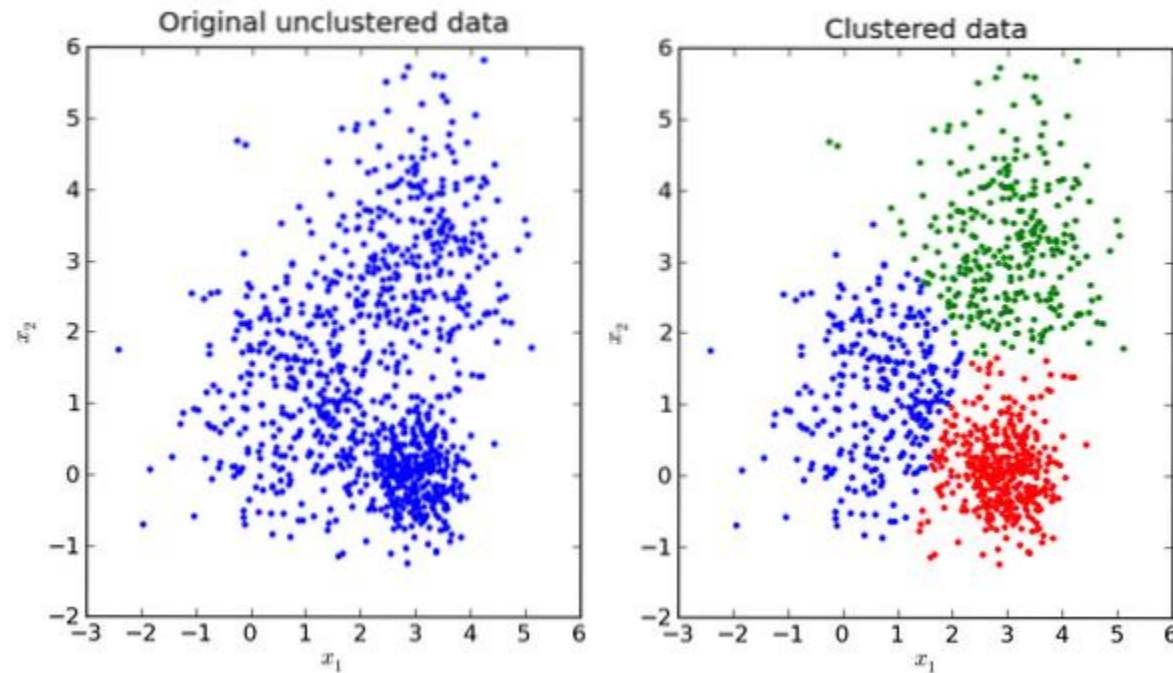


Unsupervised Machine Learning

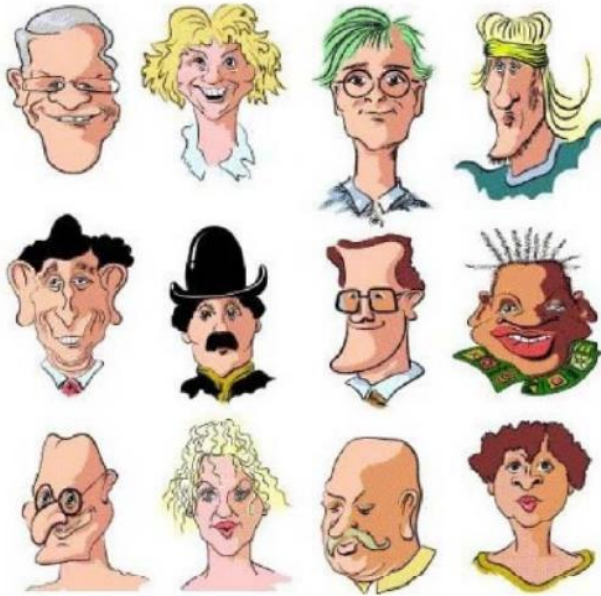


Clustering is a hard task

- Often, data cannot be easily separated (especially multi-dimensional data)



Example: group people according to their traits



- Far more complex than supervised learning
- What means “similar”?
- How many groups can we have?

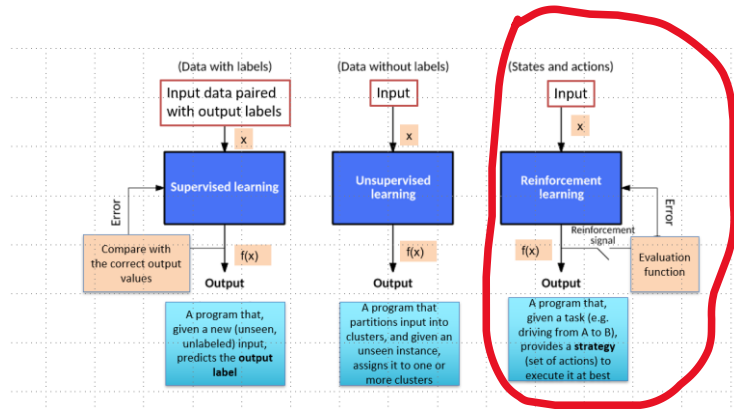
Very ill-formed problem:

There are several ways of grouping depending on the chosen similarity function
(e.g. Bold/hairy, female/male, hair color, age, etc.)

Are they similar?

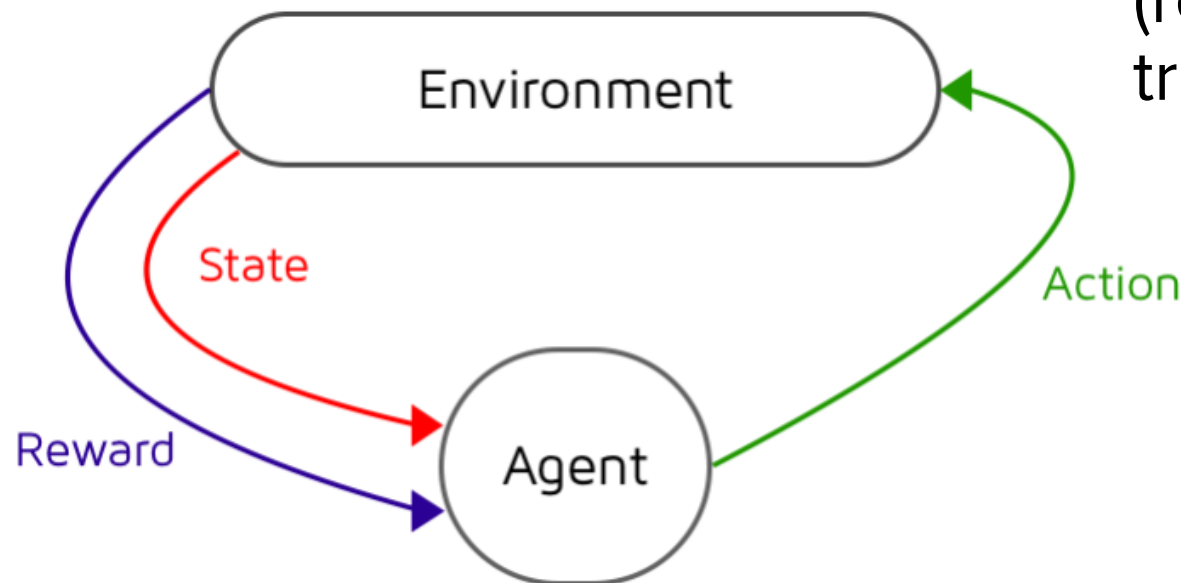


Reinforcement learning: a.k.o. “loosely” supervised

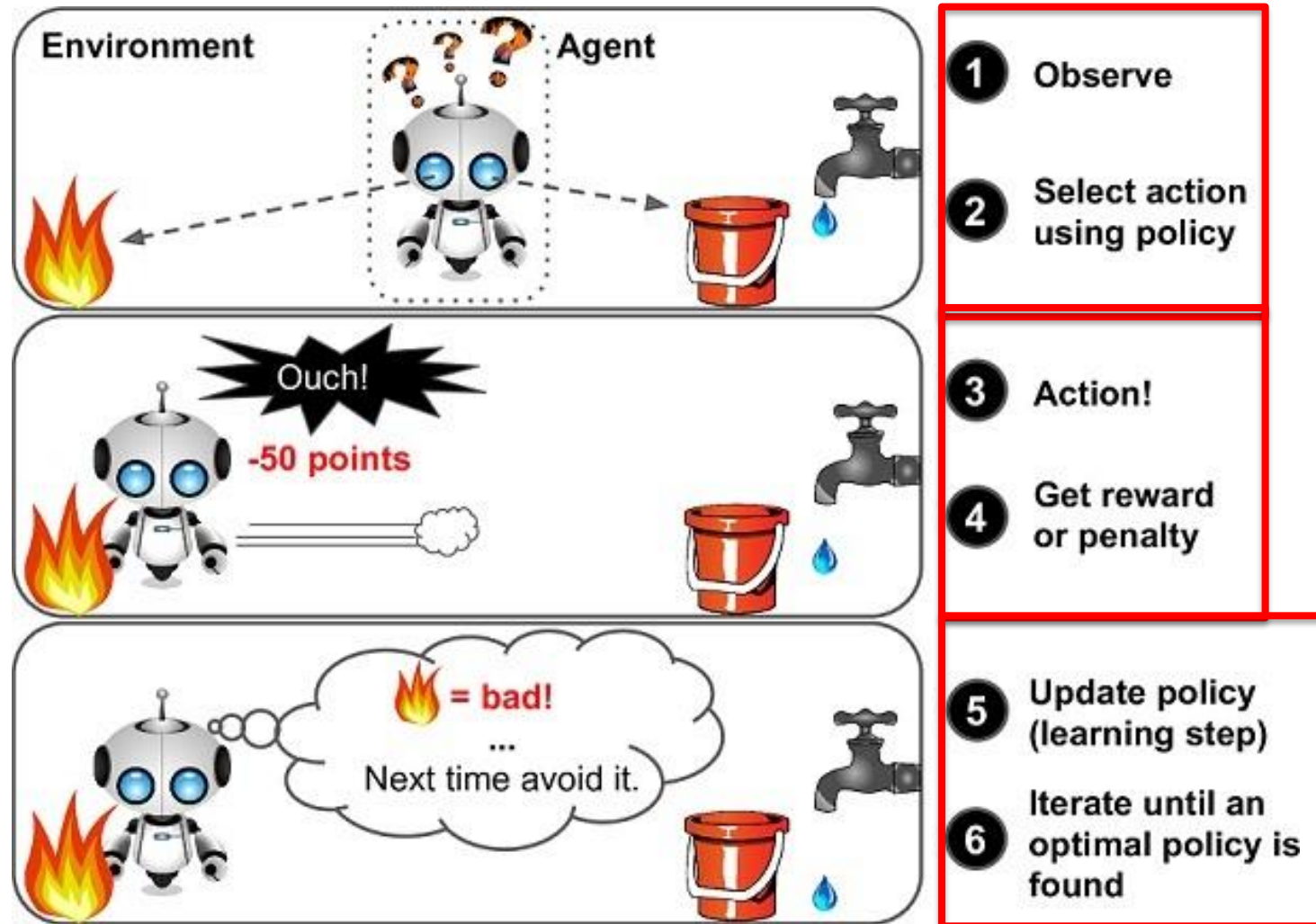


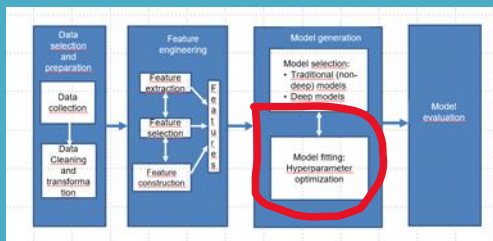
- The objective is to learn a “policy” or strategy to act in an environment

- **No examples.** Try and test
- The “agent” learns guided by rewards/penalties (rewards/penalties are a.k.o. training information)

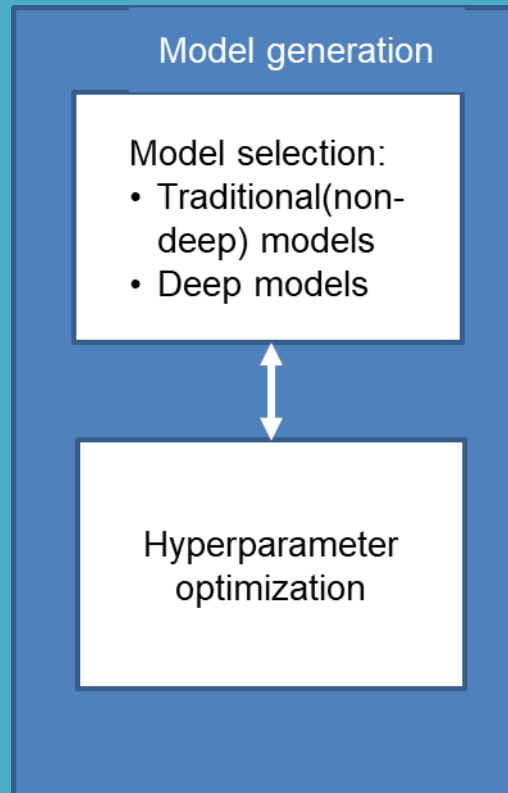


Example: robot learn to interact with dangerous environments





Parameters and Hyperparameter optimization



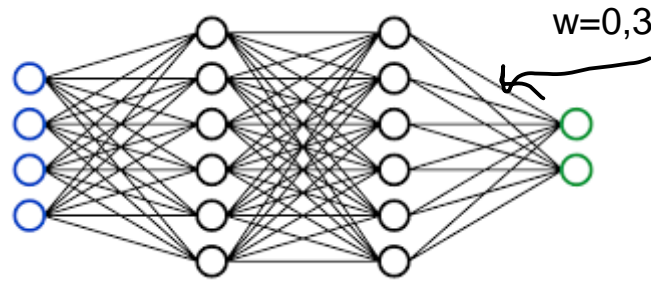
- **Learning = optimization problem** The idea is to learn a function that minimizes an error function (often called «loss») over labeled data (supervised ML), or one that maximizes some similarity function (clustering) or rewards over penalties (reinforcement).
- A model **parameter** is a configuration *variable* that is internal to the model and whose **value** can be estimated from available data during the training phase (example: the order of tested features in a decision tree).
- A model **hyperparameter** is a configuration variable that is set **prior to creating a model** (example: the number of neurons in a layer or the number of layers in a deep NN)
- Given a selected predictive algorithm (e.g., convolutional neural networks, or decision trees..), during the learning phase, **parameters** and **hyperparameters** need to be *optimized (=set to the value that produces the best performance in prediction)*
- While parameter learning is achieved during the training phase (model generation), hyperparameters must be set prior to this phase. **Different hyperparameters may lead to different models, with different performances.**

Learning: hyperparameter tuning (model fitting)

- **Hyper-parameter** is a parameter whose value must be set **before** the learning process begins. For example, the *maximum depth of a decision tree*, the number of hidden layers in a neural network, the type of loss function, *the type of similarity function* in clustering, etc.
- Hyperparameters determine **how our model is structured in the first place**
- Hyper-parameters **can significantly impact on performance**:
 - Suitable hyper-parameters must be determined for each task
 - Needed in both supervised and unsupervised learning
 - need for disciplined and [automated optimization methods](#) (greedy search, Bayesian optimization, etc..)
 - Auto-ML (meta-learning) tries to automatize this process (see, e.g. [Google AutoML](#))

Parameters and hyperparameters: an example

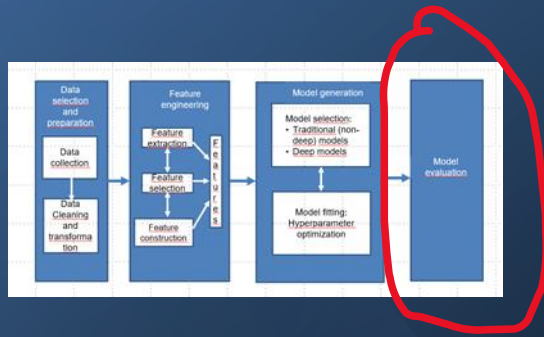
- You can think of a neural network as a multi-stage graph, where edges are directed and weighted



- The model **parameters** (to be learned) are the weights of the edges
- The model **hyperparameters** (to be set prior to the training phase) are the number of stages and the number of nodes in each stage

Common methods for hyperparameter tuning

- Of course it depends on the specific ML model and on the number of hyperparameters
- This specific topic will be analyzed only in labs
- [Grid Search and Random Search](#) are popular methods. GS it is an exhaustive search of the given hyperparameter space.
- For large number of hyperparameters, practically unfeasible with standard computational resources (other methods are used) . To learn more, read [here](#).
- Recent research area is meta-learning or [AutoML](#) (a machine able to learn automatically also its hyperparameters)



Evaluation

Model
evaluation

It is important to estimate the «goodness» of a learned model before we put it in operation

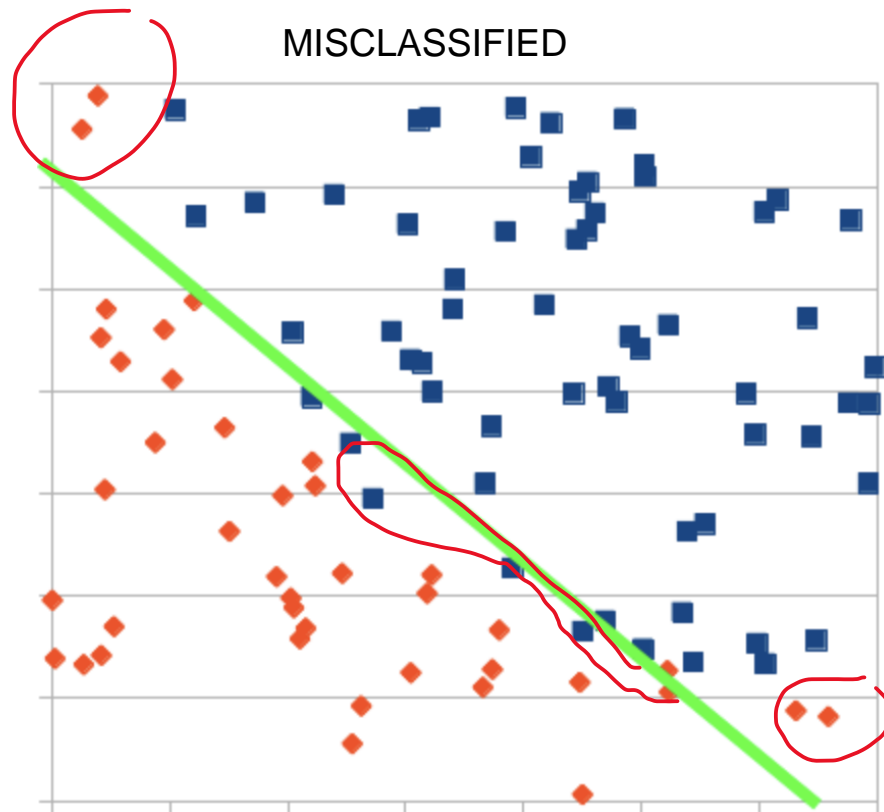
Evaluation is the process of estimating such «goodness»

The most common method is to learn the model on a fragment of the original data (called *training* or *learning set*), and evaluate its performance on a sub-sample not used during training (*test set*).

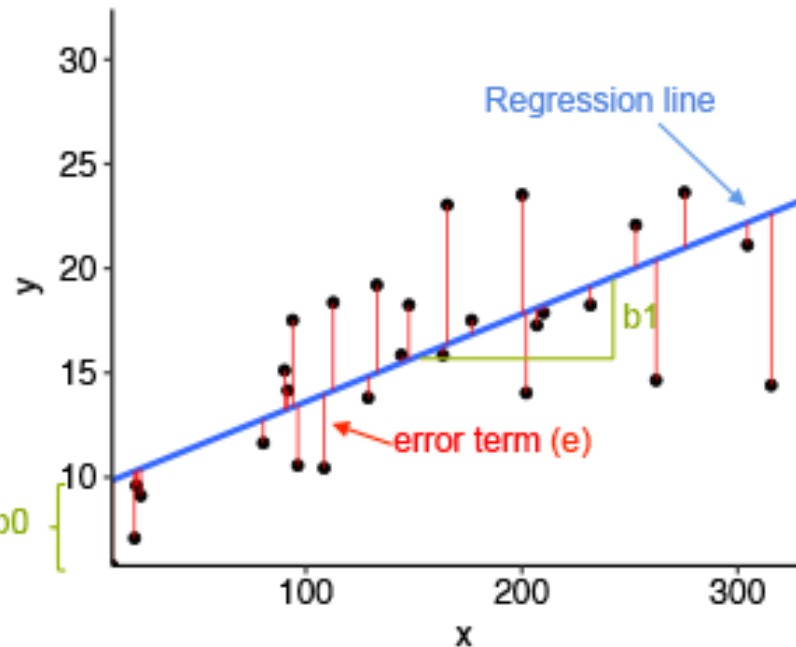
Appropriate evaluation measures are also needed (accuracy, precision, recall, AUC..)

Inappropriate measures can change the story quite a bit.. (as we will see)

In real-life applications it is impossible to learn «perfect» predictors



Classification errors



Regression errors

Classifiers may not identify perfectly the regions of each class; regressors can make errors in predicting past and future values of the real (unknown) function.

Summary

1. Look for suitable data to learn to solve a stated problem (disease gene prediction, self-driving car..)
2. Clean and preprocess the data
3. Apply feature engineering techniques
4. Choose a model (function type and algorithm)
5. Choose model hyper-parameters (parameter tuning)
6. Evaluate the performance (if not satisfactory, start again at any of previous points)